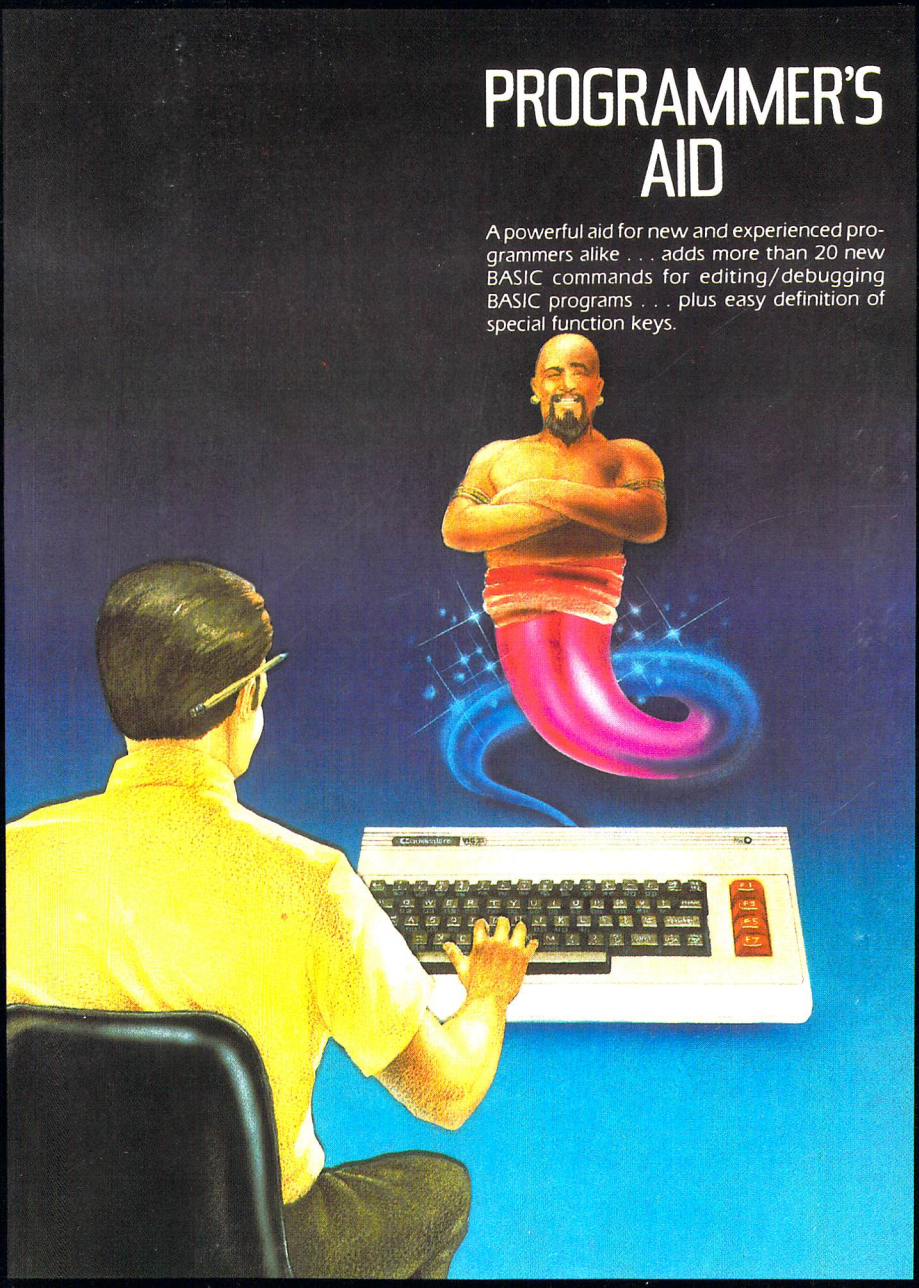
PROGRAMMER'S AID

A powerful aid for new and experienced programmers alike . . . adds more than 20 new BASIC commands for editing/debugging BASIC programs . . . plus easy definition of special function keys.



CONTENTS

INTRODUCTION 1

GETTING STARTED 2

Connecting the Cartridge 2
Programmer's Aid Commands 3
PROG and EDIT Modes 3
Using the Function 4
Commands Assigned to Function Keys 4
Redefining the Function Keys 5

NEW BASIC COMMANDS 8

BASIC Commands 8
AUTO 8
RENUMBER 10
DELETE 11
FIND 12
CHANGE 14
MERGE 15
DUMP 16
HELP 17
TRACE 19
STEP 20
OFF 22
KEY 22
EDIT 24
PROG 24
KILL 24
CONTROL FUNCTIONS 25
FUNCTION KEYS 28
NOTES 30

INTRODUCTION

The VIC 2Ø PROGRAMMER'S AID CARTRIDGE helps new and experienced programmer's write, edit, crunch and debug BASIC programs.

This cartridge adds more than 2Ø special editing commands to your VIC's instruction vocabulary and gives you an easy method for redefining the VIC's programmable function keys.

If you're an experienced programmer you already know how helpful it is to automatically number and renumber program lines, delete and insert segments, view program lines as they run, jump to commands or character strings, and use the VIC's special function keys to increase your programming power.

If you're just starting out in programming, the PROGRAMMER'S AID CARTRIDGE will make it easier to write your own programs, understand how they work and find and correct errors.

The PROGRAMMER'S AID CARTRIDGE is only one of a growing range of "programming aids" Commodore is developing to make BASIC programming easy to learn and enjoy. For more information on VIC 2Ø programming accessories, visit your Commodore dealer.

GETTING STARTED

Connecting the Cartridge

The VIC 20 PROGRAMMING AID CARTRIDGE may be inserted directly into the VIC's Expansion Port slot on the back side of the computer, and will also work with the VIC 20 Expansion Module.

Here is a step-by-step explanation of how to connect and start the cartridge:

1. Make sure the VIC 20 is turned OFF. This will greatly increase the life and performance of your cartridge.

2. Insert the cartridge into the Expansion Port slot on the back side of the VIC. Make sure the cartridge label is facing up as you insert the cartridge.

3. After the VIC is inserted, type the following:
   SYS 28681 (and hit the RETURN key)
   The VIC will display the following message:
   PROGRAMMER'S AID
   READY.

4. All of the AID commands are now included in your VIC's internal instruction set and may be used like any BASIC commands.

5. **CAUTION:** Always turn the VIC OFF before inserting or unplugging a cartridge.
   Note that the action of some machine language programs may vary when used with the PROGRAMMERS' AID.
   Do not open or touch the inside of the cartridge, which may be damaged by direct contact with static electricity.

PROGRAMMER'S AID COMMANDS

Here is a list of "AID" commands which have been added to your VIC's BASIC vocabulary:

AUTO	**FIND**	**OFF**
CHANGE	**HELP**	**PROG**
DELETE	**KEY**	**RENUMBER**
DUMP	**KILL**	**STEP**
EDIT	**MERGE**	**TRACE**

The PROGRAMMER'S AID CARTRIDGE also gives you several special program editing functions which you can use by typing the CTRL key and a letter on the keyboard. For example, if you hold down the CTRL key and type the letter N you will erase all the characters in your program after the cursor. Other editing functions are shown below:

CTRL A Scroll up a program list
CTRL E Cancels quote and insert modes
CTRL L Erase characters after curosr on same line
CTRL N Erase all characters in program after cursor
CTRL Q Scroll down a program list
CTRL U Erase all characters on cursor line

PROG AND EDIT MODES

The PROGRAMMER'S AID CARTRIDGE has TWO MODES of operation which effectively double the cartridge's editing power. Both modes let you use the VIC's programmable function keys on the right side of the keyboard to quickly enter a command by typing one key.

When you start the PROGRAMMER'S AID CARTRIDGE, the VIC is automatically in PROGRAM mode. This means the special function keys are designated with programming commands such as GOTO, CHR$(, and GOSUB.

The second mode is called EDIT mode. In this mode, the function keys are designated with program editing commands like RENUMBER, DELETE and FIND.

Note that you can type in any of the special AID commands at any time by typing the full word. You don't have to be in a special mode to use any of the commands or functions listed above. For example, if you're in PROG mode and want to use the FIND command, you can't use F5 to get that command but you can still type in the full word FIND and use

the command that way. The PROG and EDIT modes don't cut down the PROGRAMMER'S AID command set...they just assign the function keys.

Here are some important things to remember about EDIT and PROGRAM modes:

1. The primary purpose of these two modes is to assign special commands to the VIC's function keys so you can write and edit your BASIC programs faster. You can still type in any AID command at any time in either mode.
2. You are automatically in PROGRAM mode when you start the AID cartridge.
3. To switch to EDIT mode simply type the word EDIT and hit RETURN...or...hold down the CTRL key and type the F1 key, then hit RETURN. Either action will put you in EDIT mode.
4. To get back to PROGRAM mode from EDIT mode, type PROG and hit RETURN...or...hold down the CTRL key and type the F1 key, then hit RETURN.

USING THE FUNCTION KEYS

It's easy to use the VIC's function keys with the PROGRAMMER'S AID CARTRIDGE. There are four function keys but many more functions than keys. Here's how we get them -- first, if you type the keys all by themselves you get keys F1, F3, F5 and F7. If you hold down the SHIFT key and type the same keys you get F2, F4, F6 and F8. If you hold down the CTRL key and type the same keys you get F9, F1Ø, F11 and F12. (Note that F9, 1Ø, 11, 12 are not inscribed on the keys). You can also use the CTRL key and several letters on the keyboard in combination to perform special editing functions...this is accomplished by holding down the CTRL key while typing that letter (A, E, L, N, Q, U).

When you turn on the VIC and start the AID CARTRIDGE, the function keys are designated with the commands you see under PROGRAM MODE in the chart below. If you switch to EDIT mode the key designations are changed to the commands listed under EDIT MODE below.

PROGRAMMER'S AID COMMANDS ASSIGNED TO FUNCTION KEYS

The following chart shows how the function keys are defined in PROGRAM and EDIT modes, and also lists the extra editing commands you can use in the EDIT mode:

PROGRAM MODE	EDIT MODE
UnSHIFTed	*UnSHIFTed*
(KEY 1) F1...LIST	F1...LIST
(KEY 3) F3...RUN ←	F3...RUN ←
(KEY 5) F5...GOTO	F4...DELETE
(KEY 7) F7...INPUT	F7...TRACE ←
SHIFTed	*SHIFTed*
(KEY 2) F2...MID$(	F2...AUTO
(KEY 4) F4...LEFT$(	F4...DELETE
(KEY 6) F6...RIGHT$(	F6...CHANGE
(KEY 8) F8...CHR$(	F8...STEP ←
CTRL	*CTRL*
(KEY 9) CTRL F1...EDIT ←	CTRL F1...PROG ←
(KEY 10) CTRL F3...GOSUB	CTRL F3...RENUMBER
(KEY 11) CTRL F5...RETURN ←	CTRL F5...MERGE
(KEY 12) CTRL F7...STR$(	CTRL F7...OFF ←

← means automatic RETURN is entered and is obtained by typing CTRL RVS-ON keys and the back arrow).

CTRL & LETTER

All CTRL/Letter editing commands shown under PROGRAMMER'S AID COMMANDS above remain the same in both PROGRAM and EDIT modes.

REDEFINING THE FUNCTION KEYS

The PROGRAMMER'S AID CARTRIDGE lets you change the VIC's function key assignments to anything you like...a BASIC command, number, text, graphics, subroutine...as long as the information being defined is 10 characters or less in length for any one key.

Try this: Type the word KEY and hit the RETURN key. This displays a "menu" of key numbers 1-12 with the commands they represent. (The "back arrow" at the end of some lines specifies a "carriage return" which automatically executes that command"). To change a command assigned to a function key, simply CHANGE THE INFORMATION BETWEEN THE QUOTATION MARKS AND HIT RETURN. The new information can be a BASIC command, number, character string, graphics, or even a GOSUB to a subroutine. Here's an example to show you how it works:

1. Type the word KEY and hit the RETURN key
2. Use the CRSR keys to move the cursor to the "in the first line.
3. Type the word "PRINT" and hit the RETURN key.

4. Now when you hit the F1 key the word PRINT will appear on your screen. You can use F1 whenever you want to type the word PRINT, just as if you typed in the PRINT command. Try typing the F1 key and then type 2*5 and hit the RETURN key. The number 10 should appear on your screen because if you PRINT 2*5 the VIC computes the answer which is 10.
5. Now let's redefine Function Key 1 to a number. Type KEY and hit RETURN. Now change the "PRINT" in the first line to "50" and hit RETURN. Type the following:

   PRINT [F1] * 10 (and hit RETURN)
6. You have redefined F1 as the number 50 and every time you type F1 the number 50 appears. If you PRINT the sum of F1 (50) times 10 as we just did, you get the answer 500. You can redefine any function key as a number, a word, a BASIC command, graphic symbol, an operator (*/$-), or even as an entire SUBROUTINE (try defining a function key as GOSUB with a subroutine line number).

   Function keys can also help you program faster. For example, let's say you're writing a long program and you have to keep RUNning it to debug a portion near the end. This is a long-running program and you don't want to wait for the whole program to run each time out check out the short segment you're working on...so you redefine a function key as "RUN 9000" (presuming 9000 is the start of the program section you're working on). To view the portion you're editing, just type the function key and RETURN...or...build the RETURN into the command like this:

Adding Carriage Returns. If you want to get fancier, you can add a RETURN to any command by typing the following sequence:

1. Type the KEY command and hit RETURN.
2. Enter your command between the quote marks.
3. Type this sequence immediately after the command:
   a. CTRL and RVS ON (this turns on the Reverse mode)
   b. Type the BACK ARROW KEY (top left on keyboard)
   c. Type the ENDING QUOTE MARKS (") and hit RETURN.
4. Now when you type the KEY command you will automatically get a RETURN after the command as in KEY 3 which is defined as the RUN command with a carriage return built in as indicated by the REVERSE BACK ARROW.

One last point about the KEY command...if you define a key with information which is too long (longer than 10 characters) you will get an ILLEGAL QUANTITY ERROR and the VIC will only accept the first 10 characters as the key definition.

NEW BASIC COMMANDS

BASIC Commands

This chapter deals with the new BASIC commands added by the PROGRAMMER'S AID CARTRIDGE.

The format of each command is in accordance with the following method of notation:

1. The items written by the capital letters of the alphabet should be typed in exactly as shown.
2. Items in brackets show information usually specified by the user, such as the line numbers of the program you're editing. The brackets are not typed.
3. Other symbols such as the commas and double quotation symbols should be inputted exactly as shown.
4. Note that O as in Oscar is distinguished from the zero (0) to avoid confusion.

AUTO

Format: AUTO [First line number], [Interval between line numbers].

Purpose: Line number is automatically generated for each input of the carriage return.

Explanation: AUTO command indicates the line number automatically when the BASIC command is entered. When [First line number] is indicated, input the BASIC statement, then depress the RETURN key. By doing so, the value specified for [Interval of line number] will be added, and the new line number will be indicated. When [First line number] or [Interval of line number] is not specified, (the AUTO command) functions as AUTO 100 or 10. In case the AUTO has been executed in advance, the AUTO memorized the last line number and the line number interval at the time such execution made in advance.

Example: Input AUTO RETURN after typing NEW and RETURN.

AUTO RETURN (Hereinafter RETURN will be omitted)

100 ■ ← Cursor

In the case, the line number 100 will be indicated, then the cursor starts to flicker. From this point, start to input the BASIC statement. To cancel the auto mode, simply depress the RETURN key after the line number.

```
100 REM PROGRAMMERS'
110REM AID PACK
120 REM AUTO COMMAND
130
```

After the cancellation of the AUTO mode, some other actions (for example, RUN) become possible. After that, it is also possible to continue the AUTO mode from the point where the cancellation is made.

```
RUN
READY.
AUTO
130 ■
```

AUTO always memorizes the number of the last line. When you want to start from a new line number, respecify the number using AUTO [New line number], like this:

```
AUTO200
200 PRINT "PROGRAMMERS' AID PACK"
210 ■
```

To change the interval of the line number, it is necessary to specify both the [first line number] and the [distance of line number].

```
AUTO,25
?SYNTAX
ERROR
READY.
AUTO1000,25
```

Enter some lines to check if the interval of the line number is changed. Even when the first line number is changed, AUTO memorized the interval of the line number.

```
AUTO2000
2000 REM LINE STEP
2025
```

RENUMBER

Format: RENUMBER [First line number], [Interval of line number].

Purpose: To renumber the line numbers of the program.

Explanation: The RENUMBER command renumbers all the program lines that have been entered in the VIC. This command also renumbers the line numbers to specify the direction of the branching at the points for GOTO, GOSUB, IF - THEN, ON - GOTO and ON - GOSUB. If a line number for a branch operation is not given, the number 63999 is automatically assigned. For example, if you have a GOTO 15Ø and nothing on line 15Ø the "GOTO 15Ø" will be changed to "GOTO 63999" in the numbered program. The default (automatic) beginning and interval numbers will be 100 and 10 if [First line number] and [Interval of line number] are not specified.

Example: Enter the following program (Do not forget to type NEW RETURN in the beginning. Always follow this procedure when entering a new program).

```
1Ø REM RENUMBER
2Ø GOTO1Ø
3Ø GOSUB1ØØØ
4Ø IF A=1 THEN 5ØØ
1ØØØ REM SUBROUTINE
1ØØ2 RETURN
```

You type this and hit RETURN

```
RENUMBER 1ØØ,1Ø
READY
LIST
1ØØ REM RENUMBER
11Ø GOTO 1ØØ
12Ø GOSUB14Ø
13Ø IF A=1 THEN 63999
14Ø REM SUBROUTINE
15Ø RETURN
```

Notice that in this example the GOTO in line 2Ø and GOSUB in line 3Ø have been changed to correspond to the new line numbers. The direction of branching for line 4Ø is given as 63999, since there is no 5ØØth line in this program. (Note: to look for a line number which has not been entered yet, use FIND 63999 after RENUMBER).

When [Interval of line number] has to be changed, the first line number should also be changed.

Wrong

```
RENUMBER,3
?SYNTAX
ERROR
READY.
```

Right

```
RENUMBER300,3
READY.
LIST
300 REM RENUMBER
303 GOTO300
```

When the last line number has become greater than 63999 as the result of the renumbering of the line numbers, the following message will be given.

```
RENUMBER50000,10000
?OUT OF RANGE
ERROR
```

When this? OUT OF RANGE ERROR is given, the line numbers of the program will not be changed. (Try it for testing).

DELETE

Format: DELETE [Line number] - [Line number].

Purpose: To delete the line of the program.

Explanation: The line to be deleted can be specified in the same manner as in the case of a LIST command as shown:

DELETE 500	To delete line number 500
DELETE 50-	To delete from and after line number 50
DELETE -50	To delete up to line number 50
DELETE 50-500	To delete the line numbers from 50 to 500

Example: (Type this program)

```
10 PRINT"DELETE"
20 PRINT"TO DELETE LINES"
30 PRINT"[START LINE]-[END LINE]"
40 PRINT"[END LINE]"
50 PRINT"[START LINE]-"
60 PRINT"[THE LINE]"
DELETE 30-45
```

```
READY.
LIST

10 PRINT"DELETE"
20 PRINT"TO DELETE LINES"
50 PRINT"[START LINE]"-
60 PRINT"[THE LINE]"
DELETE

?SYNTAX
  ERROR
READY.

DELETE50

READY.
LIST

10 PRINT"DELETE"
20 PRINT"TO DELETE LINES"
60 PRINT"[THE LINE]"
```

FIND

Format: FIND [BASIC code], [Line number] - [Line number] FIND "[Character string]", [Line number] - [Line number]

Purpose: To find the character string that is enclosed in the special BASIC code or quotation marks, and display all the lines including such character strings.

Explanation: In case the character string to be found is not enclosed in quotation marks, it will be converted into the intermediate language, and reached later. Thus, when the FIND PRINT is entered, all the PRINT statements will be located except the PRINT #.

If you're trying to find a character string, only the portions enclosed in the quotation marks in the program can be found. Thus, in the case of a REM statement without quotation marks, the comment cannot be located as a character string.

The line to be found can be specified in the same manner as in the case of a LIST. When the FIND is used in the case of a large program, there is the possibility that part of the result cannot be included on the display screen. In such a case, the display speed can be reduced by

depressing the CTRL key, or the display can be interrupted by depressing the STOP key.

Example: First type in the following program:

```
10 REM: FIND COMMAND
20 PRINT" ♥ CRSR COLOR EXAMPLE FOR FIND"
30 A$ =" ■ E ε ◣ ◨ ↑ ← π "
40 FOR C = 1TO8
50 PRINT MID$ (A$,C,1)
60 PRINT"R ,,;(space: 22 places)
70 PRINT"R ,,;(space: 22 places)
80 NEXT C
90 PRINT" ← ";
```

Here, let's look for the BASIC code.

```
FIND A$
30 A$ = " ■ E ε ◣ ◨ ↑ ← π "
50 PRINT MID$(A$,C,1);
```

As for the line found by the FIND (command), all the lines will be displayed so you can more easily correct the program line and re-enter it correctly. The range of FIND can be specified by using the line number. Try this:

```
FIND A$,40-
50 PRINT MID$(A$,C,1);
```

A character string can be distinguished from BASIC words by enclosing the string in quotation marks. For example:

```
FIND FOR
40 FOR C = 1TO8

FIND "FOR"
20 PRINT " ♥ CRSR COLOR EXAMPLE FOR FIND"

FIND "FIND"
20 PRINT " ♥ CRSR COLOR EXAMPLE FOR FIND"
```

The FIND in line 10 cannot be found, since it is not enclosed in quotation marks.

When enclosed in quotation marks, the cursor control and the specification of character color can be detected, as shown:

```
FIND " ← "
30 A$ = " ■ E ε ◣ ◨ ↑ ← π "
90 PRINT" ← ;
```

CHANGE

Format: CHANGE [Old BASIC code], [New BASIC code], [Line number] -[Line number] CHANGE "[Old character string]" "[New character string]", [Line number] - [Line number].

Purpose: To find any given BASIC code or the character string enclosed in the quotation marks, and display it after converting it into the new BASIC code or new character string.

Explanation: Use in the same manner as that of the case of the FIND.

Example: Here, let's try the CHANGE by the program used in the case of the FIND command. Let's change A$ to B$:

```
CHANGEA$,B$
30 B$ = " ■ E ε ◣ ◨ ↑ ← π "
50 PRINT MID$(B$,C,1);
```

Next, let's change the variable C to the variable A.

```
CHANGE C,A
10 REM: FIND AOMMAND
40 FOR A = 1TO8
50 PRINT MID$(B$,A,1)
80 NEXT A
```

In this case, not only the variable C but also the C in the REM sentence have been changed. But this is an unusual case which only affects REM statements.

```
CHANGE I,O
10 REM: FOND AOMMAND
```

As with the FIND command, cursor control and the specification of character color can be changed when enclosed in quotation marks.

```
CHANGE "♥ CRSR ", "♥ CRSR CRSR "
20 PRINT "♥ CRSR CRSR COLOR EXAMPLE FOR FIND"
```

MERGE

Format: MERGE "[Program name]", [Device number]
Purpose: To load the previously saved program or subroutine, and mix it with the program currently stored in the memory of the VIC.

Explanation: The device number represents the number of the device for loading the subroutine. When not specified, the device number = 1 (cassette drive).

The program name can be specifed in the same manner as an ordinary LOAD. The pogram name can be specified either by specifying it directly in the form of "Program name" or by specifying it by a variable. When the program name is not specified, the VIC loads the first-found program for merging.

Example: Enter the following short program SAVE it.

```
10 REM: PROGRAMMER'
30 REM: AID PACK
50 REM: MERGE COMMAND
SAVE"MERGE"
```

Rewind the tape and delete this program, then enter the following program:

```
20 REM: "PROGRAMMERS"
40 REM: "AID PACK"
60 REM: "MERGE COMMAND"
```

Then, execute the MERGE command.

```
MERGE "MERGE"

PRESS PLAY ON TAPE
OK

SEARCHING FOR MERGE
FOUND MERGE

LOADING
READY.
LIST

10 REM: PROGRAMMERS'
```

```
2Ø REM: ''PROGRAMMERS'''
3Ø REM: AID PACK
4Ø REM: ''AID PACK''
5Ø REM: MERGE COMMAND
6Ø REM: ''MERGE COMMAND''
```

This is an extreme example. In general, the MERGE command is used when mixing or adding one program or subroutine to another.
Merging by means of variables is also possible.
For example, when inputted as

```
X$ = ''VIC''
MERGE X$
```

the program named VIC can be merged.

CAUTION: In case the same line numbers are present in the two programs to be merged, they will be displayed in parallel to each other, but such case should be avoided when possible.

DUMP

Format: DUMP

Purpose: To display all the variables (except the array variables) defined in the memory at the time of the execution of the DUMP (command).

Explanation: The variable will be displayed in the form of the name of variable = value. The value of variable can be changed by the screen editor.

When there are a number of variables in the program, the screen cannot cover all of these variables, so that the variables defined earlier will disappear from the screen because of the scrolling of the display. In such case, the DUMP can be stopped temporarily by depressing the CTRL key or SHIFT key. When the STOP key is depressed, DUMP, will be finished. To let the variables to be corrected remain on the screen, use the SHIFT key (or CTRL key) in combination with the STOP key.

The variables will be displayed in the order of their definitions.

Example: Input the following program:

```
1Ø REM: RANDOM COLOR
2Ø PRINTCHR''♥''
```

```
30 SC = INT (RND (8) *15) + 1
40 BC = INT (RND (8) *7 + 1
50 R = SC*16 + 8 + BC
60 POKE 36879,R
70 FOR I = 1TO500:NEXT
80 GOTO20
```

Execute the RUN (command), the depress the STOP key.

```
BREAK IN 70
READY.

DUMP
SC =  5
BC =  6
R =  94
I =  89
```

The DUMP command always displays the value of the variable at the time when it is executed.

HELP

Format: HELP

Purpose: To display not only the line including the error but also indicate the location of the error with reverse characters on the screen when the error is detected.

Explanation: The HELP command must be used immediately after the error message, otherwise the VIC will not locate the location of the error. Unlike the case where an error is detected by the VIC, when the operator (User) has stopped execution of a program with the STOP key, the HELP command displays the line including that part of the statement which was last executed properly.

Example: Input the following program as it is.

```
10 FOR D = -180TO180 STEP20
20 Y = SIN(D π /180)                    [π = pi sign]
30 I = 10*Y + 11
40 PRINT TAB(I); "*"
50 NEXT X
```

Execute the RUN (command),

```
?SYNTAX
  ERROR IN 2Ø
READY.
```

As a result, the above error message will be given. In this case, when the HELP command is used, the following display will be given.

```
HELP
2Ø Y = SIN(D π /18Ø)
READY.
```

And D will be shown by the reverse characters. When observed carefully, we can that * is not present between D and π. As such, in most instances, HELP indicates the character located before the place of error. The error can be corrected easily by the screen editor as usual.
HELP must be used immediately after the error message is given. However simple the command is, when it is used before HELP, the memory of the error in the VIC becomes ineffective. Run the corrected program.

```
?NEXT WITHOUT FOR
  ERROR IN 5Ø
READY.
LIST5Ø

5Ø NEXT X
HELP

READY.
```

In this case, the VIC does not ''remember'' the location of the error. Run the program again.

```
?NEXT WITHOUT FOR
  ERROR IN 5Ø
READY.
HELP
5Ø NEXT
```

X is shown by a reverse character. Checking the corresponding FOR

by the LIST, we can notice that the loop variable has become D. To execute the program, correct X to D.

TRACE

Format: TRACE

Purpose: To display the line number currently executed.

Explanation: When the TRACE command is given, and the program is run, the numbers of the six lines which have been executed will be displayed in the "Window" appearing on the upper right of the screen. Each time when the line is executed, the line number displayed in the "Window" will be scrolled up so that the latest-executed line number will always appear last. Also note that the TRACE command tends to slow down the execution speed of the program considerably.
To reduce the speed of the TRACE, depress either the CTRL key or SHIFT key. By doing so, the speed of TRACE can be reduced to a speed of about 2 lines per second.

```
10 PRINT"♥"
20 A=0
30 A=A+1
40 PRINT" S "A
50 GOTO30
```

Then, give the TRACE command.

```
TRACE

READY.
RUN
```

Display on the screen will be erased, and the "Window" of the reverse field appears on the upper right corner of the screen. The line numbers appear in the upper left corner of the screen, and increase by one.

```
#30
#40
#50
#30
#40
#50
```

In the "Window", the numerical character preceded by the # symbol represents the number of the executed line.

To stop the program, depress the STOP key. Note, however, that the STOP key will not work when the SHIFT key has been depressed.

After stopping the TRACE of this program, depress the RETURN key, then you will notice the appearance of ? SYNTAX ERROR. When the RETURN key is depressed, the VIC scans the line on which the cursor is placed. In the case, the cursor is on the fifth line when you stop the run of the program.

Since the "Window" still remains on the right side (of the screen), it can be considered that the VIC assumes that # (Line number) has been inputted. If the cursor is in the above six lines at the time when the TRACE command is cancelled, proceed to the next action after either putting the cursor down or erasing the display on the screen.

Check how the TRACE command works at the time when the program is looped to one of the lines.

TRACE the following program:

```
10 PRINT"■"A:A = A + 1:GOTO10
```

When this program is traced, only #10 will appear in the "Window", and it stops at the bottom. When the program is looped on one of the lines, there will be no chance that the "Window" becomes full of the line numbers. In many programs, "GET loop" as shown below is used.

```
120 GETA$: IF A$ = "" THEN120
```

Since the TRACE will not display #120 repeatedly in the window at the time of such looping, the TRACE can be performed easily.

To cancel the TRACE mode, use the OFF command.

STEP

Format: STEP

Purpose: To stop (the run of program) after its one line has been executed. When the SHIFT key is depressed, the next line will be executed. The line number will be displayed in the "Window" on the upper right of the screen.

Explanation: When the STEP is used, the lines of the BASIC program will be executed one by one. The number of the line to be executed

will be displayed at the bottom of the "Window" on the upper right of the screen.

To execute the next line, depress either the SHIFT key or CTRL key. When the SHIFT key (or CTRL key) is kept depressed, the lines will be executed continuously and one by one until the key is released.

To stop the run of the program, depress the STOP key.

After the RUN (of the program), the SHIFT key must be depressed even when executing the first line. In case there is an infinite loop such as

```
10Ø GOTO10Ø
```

in the program, this line will continue to execute. To release the loop, depress the STOP key.

To cancel the STEP mode, use the OFF command.

Example: Input the following program, and execute the STEP command.

```
1Ø PRINT"HOP"
2Ø PRINT"STEP"
3Ø PRINT"JUMP"
4Ø GOTO1Ø
STEP

READY.
RUN
```

At this point, the window appears on the upper right corner of the screen of the VIC, and line number #10 will be displayed at the bottom of the window. Note, however, that the 10th line had not been executed yet at this point.

When the SHIFT key is pressed, HOP and STEP will be displayed, and #1Ø, #2Ø, #3Ø will be displayed in the window. Each time when the SHIFT key is depressed, one line of the BASIC program will be executed, and the number of the statement to be executed next will be displayed. When the SHIFT key is kept depressed, the line numbers in the window will be scrolled just like the case of the TRACE.

When the STOP key is depressed, the run of the program will be suspended. When the STOP key is pressed while the cursor is at the bottom of the screen, only the last two line numbers can be seen in the window, because the display on the screen has been scrolled in order for the VIC to display the messages of BREAK IN XXX and READY.

When encountered with the program to be looped in one line, the loop will be executed continuously until the STOP key is depressed. Such a line number, however, stands still at the bottom of the window.

OFF

Format: OFF

Purpose: To cancel the TRACE mode and STEP mode.

Explanation: The functions of TRACE and STEP can be cancelled by using the OFF command. When the OFF command is given, the "Window" will not appear even when the program is run, and it is not necessary to depress the SHIFT key (or CTRL key) before executing the next line.

Example: Run the program used for the explanation of the STEP mode while the STEP is at work. Then, suspend the execution of the program by depressing the STOP key. Then, type as OFF after clearing the screen. Subsequently, run the program again, then you will notice that the window will not appear this time, so that the program can be executed as usual.

KEY

Format 1: KEY

Format 2: KEY [Key number], "[Character string]". Note, however, that the key numbers are 1-12, and the character string is up to 1Ø characters.

Purpose: To display the command (character string) assigned to the function key. Also, to change the command (character string) assigned to each function key to other command (character string).

Explanation: There are two modes for the group of the commands to be assigned to the function key by the programmer's aid pack (Refer to p. 4). These are the PROGRAM mode and EDIT mode. After the pro-grammer's aid pack is initialized, the 12 commands of the program's mode are first assigned. In this case, the following key list will be displayed when the key command is inputted.

```
KEY
KEY 1,"LIST"
KEY 2,"MID$  "
KEY 3,"RUN ←"
```

```
KEY 4,"LEFT$  "
KEY 5,"GOTO"
KEY 6,"RIGHT$  "
KEY 7,"INPUT"
KEY 8,"CHR$  "
KEY 9,"EDIT←"
KEY 1Ø,"GOSUB"
KEY 11,"RETURN← "
KEY 12,"STR$  "
```

When the f 1 key is depressed while also depressing the CTRL key, the EDIT (← is the symbol of RETURN code) of KEY 9 will be executed, and the entry into the EDIT mode will be accomplished.
In the case of the EDIT mode, the following key list will be displayed when the KEY RETURN is inputted.

```
KEY
KEY 1,"LIST"
KEY 2,"AUTO"
KEY 3,"RUN← "
KEY 4,"DELETE"
KEY 5,"FIND"
KEY 6,"CHANGE"
KEY 7,"TRACE←"
KEY 8,"STEP←"
KEY 9,"PROG←"
KEY 1Ø,"RENUMBER"
KEY 11,"MERGE"
KEY 12,"OFF← "
```

When the command name is changed by the screen editor on the screen on which the key list is displayed, the new command will be assigned the to key concerned. Any character string not limited to that of command can be assigned to each key, but the character string is up to 1Ø characters. Thus, when the character strings longer than 1Ø characters is tried to be assigned, the ILLEGAL QUANTITY ERROR will be displayed.
To change the command (character string), the RETURN code or CLR code can be added readily. For example, when changing the LIST of KEY 1 to the LIST←, type as:

```
KEY 1,"LIST" + CHR$ (13)
```

or type as:

KEY 1,"LIST

Then, make the character mode reverse by depressing the CTRL key and R key, or add by typing ← key. (After this, cancel the reverse mode by the CTRL Ø). In both of the cases, when the KEY RETURN is executed, the display will be made as:

KEY 1, LIST" ← "

whereby the LIST RETURN will be exectued by depressing f 1 key. When desired to include the function of the CLR key, add the ♥ or CLR code likewise.

EDIT

Format: EDIT

Purpose: To set the function key to the EDIT mode.

Explanation: Refer to pg. 4 and KEY.

PROG

Format: PROG

Purpose: To set the function to the PROG mode.

Explanation: Refer to pg. 4 and KEY.

KILL

Format: KILL

Purpose: To cancel the function of the programmer's aid pack.

Explanation: Type as KILL, then input any one of the commmands of pages 8-24. The display will be made as:

```
?SYNTAX
  ERROR
```

When the KILL is inputted, the function of the programmer's aid pack will be cancelled.

The programmer's aid pack slows down the execution speed of the program (due to memorizing the execution position of the program by

the HELP command etc.). Thus, it is recommended to use the KILL command when the program is completed.

CONTROL FUNCTIONS

The following control functions can be made available by using the programmer's aid pack.

CTRL L

Operation: Depress the L key while depressing the CTRL (control) key.

Function: To erase all the characters after the cursor as to the line number with the cursor.

Example:

```
1Ø REM: CTRL FUNCTION
2Ø REM: [CTRL]L
3Ø REM: [CTRL]N
4Ø REM: [CTRL]U
```

Place the cursor on the E of the REM of the 20th line, and depress the CTRL key and L key. As a result, the following display will be given.

```
1Ø REM: CTRL FUNCTION
2Ø R ■ ← CURSOR
3Ø REM: [CTRL]N
4Ø REM: [CTRL]U
```

(Note) This function is for erasing only the characters displayed on the screen. For testing, take the LIST.

CTRL N

Operation: Depress N key while depressing CTRL (control) key.

Function: To erase all the characters after the cursor.

Example:

```
1Ø REM: CTRL FUNCTION
2Ø REM: [CTRL]L
3Ø REM: [CTRL]N
4Ø REM: [CTRL]U
```

Place the cursor on the E of the REM of the 30th line, and depress CTRL key and N key. As a result, the following display will be given.

```
1Ø REM: CTRL FUNCTION
2Ø REM: [CTRL]L
3Ø R■←CURSOR
```

(Note) This function is also for erasing only the characters and graphs displayed on the screen, and differs from the DELETE function.

CTRL U

Operation: Depress the U key while depressing CTRL (control) key.

Function: To erase all the characters of the line where the cursor is located. The cursor returns to the left end.

Example:

```
1Ø REM: CTRL FUNCTION
2Ø REM: [CTRL]L
3Ø REM: [CTRL]N
4Ø REM: [CTRL]U
5Ø REM: [CTRL]E
```

Place the cursor on the E of the REM of the 40th line, and depress the CTRL key. As a result, the following display will be given.

```
1Ø REM: CTRL FUNCTION
2Ø REM: [CTRL]L
3Ø REM: [CTRL]N
■← CURSOR
5Ø REM: [CTRL]E
```

CTRL E

Operation: Depress E key while depressing the CTRL (control) key.

Function: To cancel the quote mode and insert mode.

Explanation: In the case of the CBM BASIC V2 of VIC-2Ø, when the quotation marks ('') are inputted, the system enters the quote mode, and the primary function of the cursor control key becomes in-effective, and the reverse cursor control character will be displayed. Normally, to cancel the quote mode, either input the quotation mark key or depress the RETURN key. The same result as that of the above- mentioned case occurs when the cursor control key is tried to be operated after taking some blank spaces for the insertions of the characters. Such state is called the insert mode. To cancel the insert mode,

depressing the RETURN key is the only way. The quote mode and insert mode sometimes involve rather complicated procedure, and the function of CTRL E serves for the easy cancellation of these modes.

Example:

```
1Ø REM: CTRL FUNCTION
2Ø REM: [CTRL][E]
3Ø PRINT "PROGRAM"
```

In such case where the operator has noticed that the has missed to type 0 of the word PROGRAM after typing the quotation marks ('') in the 30th line, and tried to move the cursor towards the left by depressing the CRSR , the reverse (cursor control character) will be displayed, and no further action from this point is possible. In such case, the quote mode can be cancelled by depressing the CTRL key and E key, and the cursor can be moved leftward.

CTRL Q

Operation: Depress Q key while depressing the CTRL (control) key.

Function: To scroll down the BASIC program list.

Explanation: In case the BASIC program consists of a long list that is too long to be covered by the display tube at a time, the scroll down function is used to take a look at the line numbers which have already passed the display tube. When the first line number is reached by the scroll down (Reverse scrolling), no further scroll down will take place. CTRL E is LIST [Line number] - [Line number], and works even when the range of the lit is limited.

CTRL A

Operation: Depress A key while depressing CTRL (control) key.

Function: To scroll up the BASIC program list.

Explanation: This is the reverse function to that of CTRL Q. This function is used to look at the later lines of a long program list. When desired to stop the scrolling of the list, release the key. When the last line is reached by scrolling up the list, no further scroll up will take place. CTRL A is LIST [Line number] - [Line number]. It works effectively even when the range of the list is limited.

FUNCTION KEYS

There are two modes for the function (command) to be assigned to the function key by the programmer's aid pack. These are the PROGRAM mode and the EDIT mode. At the time of starting, the PROGRAM mode will be selected.

The commands to be assigned to the various function keys in the PROGRAM mode are as follows:

Normal

F 1(KEY 1) :LIST
F 3(KEY 3) :RUN↵
F 5(KEY 5) :GOTO
F 7(KEY 7) :INPUT

Shift

F 2(KEY 2) :MID$
F 4(KEY 4) :LEFT$
F 6(KEY 6) :RIGHT$
F 8(KEY 8) :CHR$

Control Shift

(KEY 9) :EDIT↵
(KEY 1Ø) :GOSUB
(KEY 11) :RETURN↵
(KEY 12) :STR$

(Here, the control shift means to depress the function key while depressing the CTRL key). When each function key is depressed, this brings about the result which is the same as that brought about when the defined character is inputted by the key (↵ denotes the symbol of the RETURN code). When the CTRL key and f 1 key are depressed simultaneously, the mode will be changed to the EDIT mode. The commands to be assigned to various keys in the EDIT mode are as follows:

Normal

F 1(KEY 1) :LIST
F 3(KEY 3) :RUN↵
F 5(KEY 5) :FIND
F 7(KEY 7) :TRACE↵

Shift

F 2(KEY 2) :AUTO
F 4(KEY 4) :DELETE
F 6(KEY 6) :CHANGE
F 8(KEY 8) :STEP ←

Control Shift

(KEY 9) :PROG ←
(KEY1Ø) :RENUMBER
(KEY11) :MERGE
(KEY12) :OFF ←

As for the display of the key list of the function key, and how to change the command (character string) to be assigned, refer to KEY.

NOTES

NOTES

NOTES

Try some of these other exciting Commodore games:

Avenger
Super Slot
Super Alien
Jupiter Lander
Draw Poker
Midnight Drive
Radar Rat Race
Sky is Falling
Mole Attack
Raid on Ft. Knox
Adventureland
Pirate Cove
Mission Impossible
The Count
Voodoo Castle
Sargon Chess II
Super Smash
Cosmic Cruncher
Gorf
Omega Race
Money Wars
Personal Finance
Visible Solar System
Clowns
Bingo Math
Seawolf

Contact a Commodore retailer in your area for any of these exciting arcade style games.

commodore COMPUTER

© 1982 Commodore Electronics Ltd.